Piotr Rybarczyk

Nr Indeksu 10117

Nr Grupy Z507

Projekt: [GitHub](https://github.com/Argonus/codebook/tree/databases-2)

Projekt Zaliczeniowy

Bazy Danych II

Baza danych dla sieci restauracji

[**Wywiad z Klientem 3**](#_c51kzbpwkt8z)

[**Schemat Bazy Danych 4**](#_icy9d2pjv90q)

[Konfiguracja Globalna 4](#_si6n6yxnra6g)

[Restauracje 5](#_hbm7n6zhghx2)

[Pracownicy 6](#_ha1qflhsfdnk)

[Zasoby 7](#_i2nf6b20y749)

[Klienci 8](#_wd3y7cmw918t)

[Zamówienia 9](#_a3l53yovgspf)

[**Założenia Projektowe 10**](#_gxjq89to71ee)

[Polepszenie bezpieczeństwa bazy danych 10](#_sinfjq9iy3ef)

[Spójność Danych 11](#_e11jrkpo9x3z)

[Raportowa Baza Danych 12](#_5hdlwdetwpk0)

[**Scenariusze I Rozwiazanie 13**](#_47i0np49g7ou)

[Tworzenie kont i przypisanie ról systemowych 13](#_gclemkxxq84l)

[Tworzenie kont i przypisanie ról użytkowników 15](#_c3fygafsqi7n)

[Walidacja Danych Zamówień 16](#_r7snaiyj6mzn)

[Mechanizm Soft Delete Zamówień 18](#_40m4bz7k7tdk)

[Mechanizm Anonimizowania Danych Klienta 20](#_d0v8zumu9e6v)

[Widoki bazodanowe 24](#_grb5y7kf1ugl)

[Raportowa Baza Danych 26](#_n7iluvrm1ubb)

[**Podsumowanie i wnioski 29**](#_gjppw4drw9kg)

[**Skrypt 30**](#_isxd189pgmo3)

# Wywiad z Klientem

Firma naszego klienta przechodzi dynamiczny rozwój i koncentrując się na głównym celu, jakim jest dostarczanie posiłków 24/7/365. Aby sprostać rosnącym wymaganiom rynku i konkurencji, firma zatrudniła zespół programistów, którzy mają za zadanie przygotować nowe warstwy aplikacji i rozwinąć aktualnie istniejące a następnie zajmie się ich utrzymaniem. Celem jest umożliwienie różnym członkom zespołu pracy nad projektami niezależnie od siebie. Do nas należy, przygotowanie bazy danych tak, aby dane były bezpieczne, dostępne a sama baza skalowalna.

Firma klienta planuje wprowadzić trzy główne aplikacje:

* **Panel Administracyjny:** Zapewnia możliwość edycji wszystkich danych w systemie, z dostępem ograniczonym do nielicznych uprawnionych użytkowników.
* **Panel Restauracji:** Umożliwia edycję menu, posiłków, produktów, zarządzanie zamówieniami oraz godzinami pracy pracowników.
* **Panel Zamówień:** Służy do zarządzania zamówieniami klientów.

Dodatkowo, planowane jest wprowadzenie niezależnego **panelu raportowego**, który pozwoli na przegląd najważniejszych raportów dotyczących działalności sieci restauracji klienta. Tym zadaniem zajmą się analitycy naszego klienta, ale najpierw musimy im przygotować bazę danych na której mogą pracować.

Dodatkowo, co istotne, **schemat aktualnej baza danych nie może się zmienić**, ani stracić żadnych zawartych w niej danych. Jest już wdrożona na produkcji, i tak ma zostać.

# 

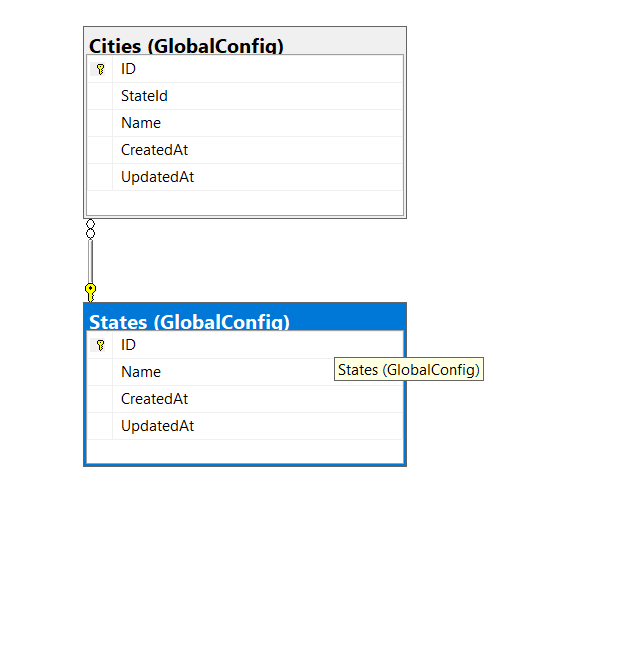
# Schemat Bazy Danych

Baza danych klienta została podzielona na wiele podobszarów, schematów, które reprezentują różne części systemu.

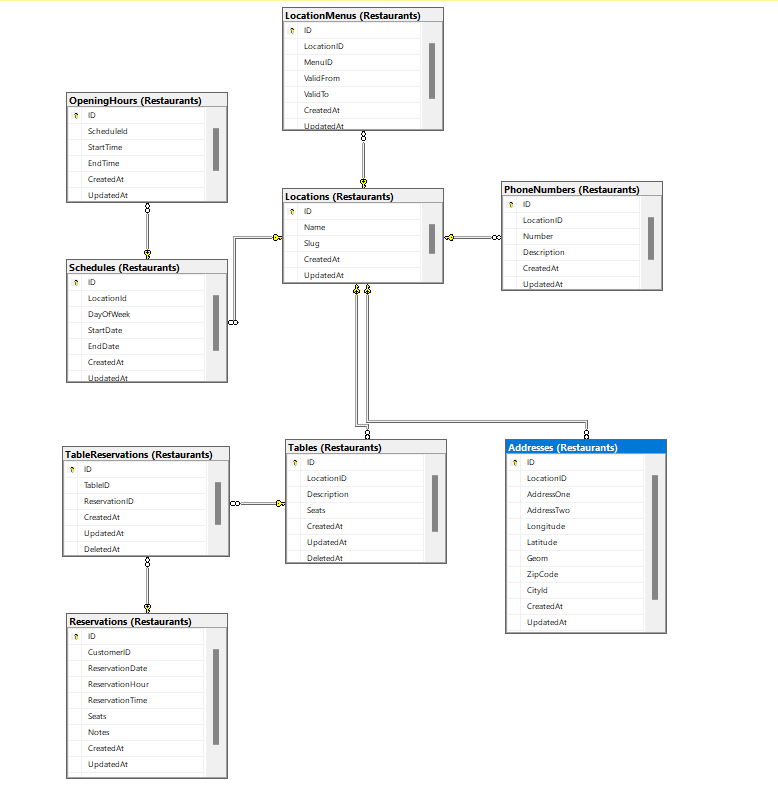
W tej części, możemy znaleźć schemat bazy danych, w dniu w ktorym otrzymujemy ja od klienta.

## Konfiguracja Globalna

Konfiguracja globalna zawiera dane, które powinny być unikalne nie zależnie od obszaru, w którym beda uzywane. Na dany moment, mamy tutaj dane geolokacyjne



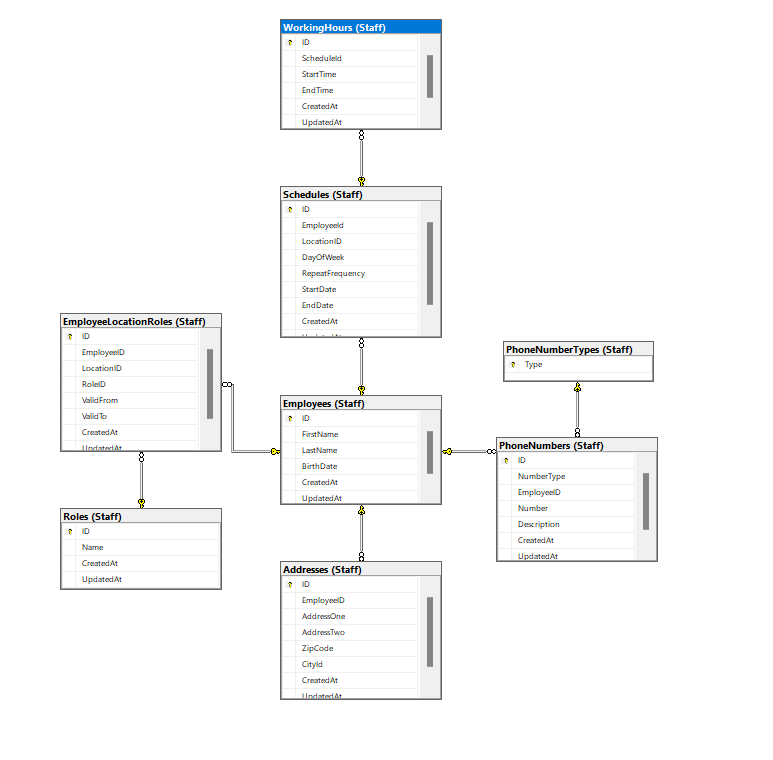
## Restauracje

Restauracje zawierają wszystkie podstawowe dane potrzebne do zarządzania restauracjami, takie jak godziny otwarcia, stoliki, dane kontaktowe ale także tabele łączące z innymi obszarami.

## 

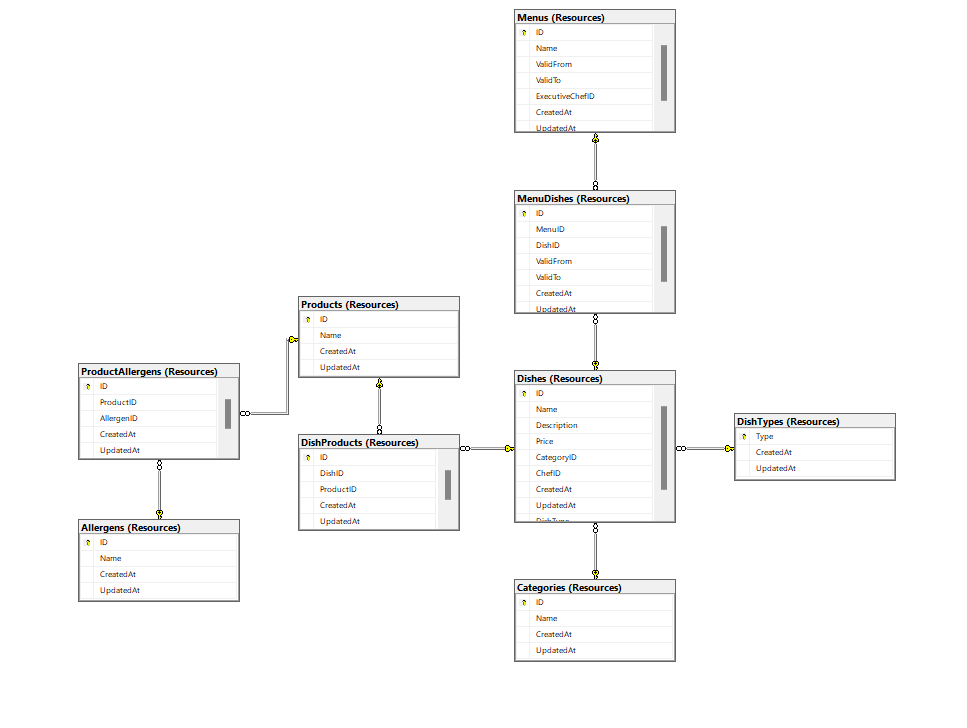
## Pracownicy

Pracownicy zawierają wszystkie dane pracowników, ale także ich rolę, godziny pracy czy dane kontaktowe.



## Zasoby

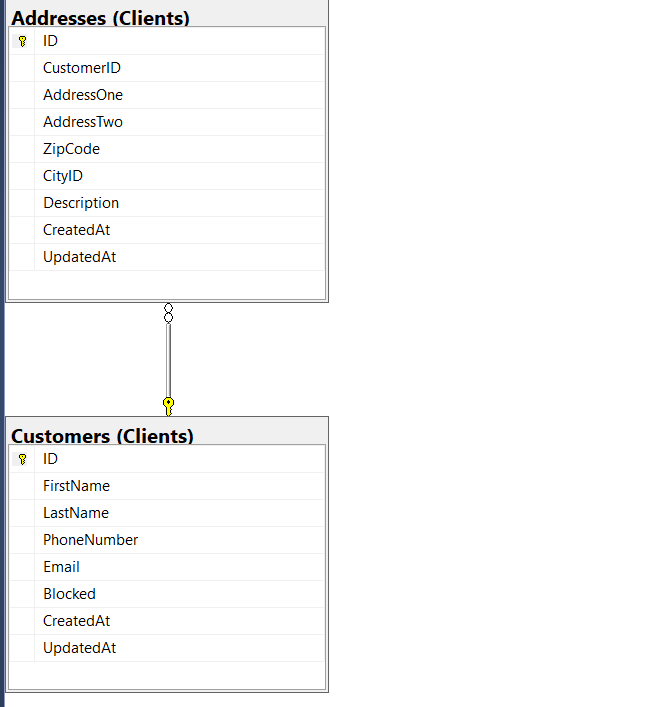
Zasoby zawierają wszystkie dane o dostępnych daniach w restauracji, wymaganych składnikach i zawartych w nich alergenach.



## 

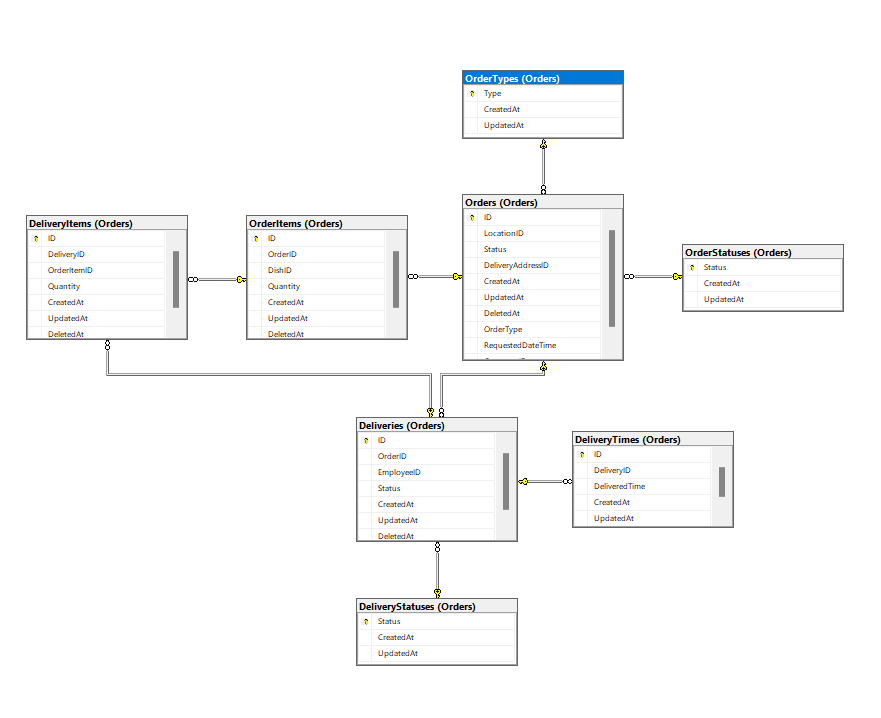
## Klienci

Klienci zawieraja wszystkie dane o klientach, ich adresy czy numer kontaktowy.



## Zamówienia

Zamówienia zawierają historię zamówień, co zostało zamówione i gdzie ale także czy już zostało dostarczone.



# Założenia Projektowe

## Polepszenie bezpieczeństwa bazy danych

Jednym z głównych celów naszego projektu, jest polepszenie bezpieczeństwa bazy danych. Wiemy, że klient ma zespół deweloperski, który ma przygotować moduły aplikacji odpowiedzialne za różne części firmy. Na tej bazie możemy przygotować zestaw odpowiednich rol i dostępów.

W uproszczeniu, można założyć, że każda rola może czytać dowolne obiekty z dowolnego obszaru bazy danych, jednak edycja czy też usuwanie danych powinno być ograniczone.

Dodatkowo, żadna z aplikacji nie może mieć możliwości zmieniania ustawień same bazy danych.

Sami deweloperzy również powinni mieć możliwość czytania danych w bazie i o bazie ale nie powinni mieć możliwości zmieniania czegokolwiek, z wyłączeniem zespołu odpowiedzialnego za maintenance. .

Na tej zasadzie możemy określić poniższe zestawy uprawnień

**Role Aplikacji:**

* superuser - specjalnie uprawnienia dla aplikacji do definiowania nowych tabel.
* administrator - uprawnienia dla konta z którego będzie korzystać aplikacje admina. Ma możliwość pracy ze wszystkimi tabelami
* restauracja - uprawnienia dla konta z którego będzie korzystać aplikacja restauracji. Ma możliwość czytania wszystkich danych i edytowania wszystkich danych.
* zamówienia - uprawnienia dla konta z którego będzie korzystać aplikacja zamówień. Może ona czytać wszystkie dane z wyłączeniem pracowników, i edytować tylko dane zamówień i klientow.
* raporty - uprawnienia dla konta modułu raportowego. Ma możliwość czytania wszystkich danych, ale nic nie może edytować.

**Role Użytkowników:**

* developer - uprawnienia dla dewelopera. Ma możliwość czytania wszystkich danych z wyłączeniem danych osobowych takie jak adresy, imiona i nazwiska
* maintenance - uprawnienia dla wybranych deweloperów. Ma możliwość czytania, edycji i usuwania wszystkich danych.

## 

## Spójność Danych

Kolejnym, istotnym dla działania serwisu naszego klienta podpunktem, jest zapewnienie spójności danych w bazie. Oznacza to, że musimy upewnić się czy typ wprowadzanych danych spełnia wymogi biznesowe. A także, że można przeprowadzać operacje opierające się na wielu tabelach jednocześnie, w sposób, który zapewni spójność danych pomiędzy nimi.

**Spójność zapisu danych**

* Powinniśmy zapewnić, że dane wprowadzane w obszarach takich jak Zamówienia czy Zasoby spełniają logiczne założenia takie jak brak negatywnych wartości w polach Monetarnych czy Ilościowych.
* Niektóre dane nigdy nie powinny być usuwane, oznacza to, że powinniśmy wprowadzić do nich mechanizm, który ukryje je przed wyszukiwaniem.
* Dodatkowo, dane personalne pracowników i klientów powinny być traktowane ze szczególną uwagą ze względu na prawa GDPR, kiedy klient poprosi o usunięcie
* konta musimy upewnić się że dane osobowe zostaną poprawnie usunięte.

**Spójność odczytu danych**

* W bazie danych znajdują się dziesiątki tabel, każda mającą konkretne znaczenie i przeznaczenie. Niektóre zapytania jednak będą się powtarzać między nimi.   
  W takim przypadku potrzebujemy wprowadzić widoki, których założeniem jest ukrycie logiki przed użytkownikiem i zapewnienie spójnej definicji danych.

## Raportowa Baza Danych

Ostatnia częścią projektu, będzie utworzenie oddzielnej bazy raportowej działającej w modelu ELT, tj. Extract Load Transform. Oznacza, to że w pierwszej części naszego projektu musimy zagwarantować przepływ i replikację danych pomiędzy bazami FoodCourt i Food Court Reports.

W procesie ELT, najlepsze praktyki mówią o utworzeniu wielu warstw logicznych w bazie raportowej tj:

* **ING\_** - pierwsza warstwa ingestion, gdzie trafiają dane nieobrobione z głównej bazy danych.
* **CLN\_** - druga warstwa, cleaned, gdzie dane zostają oczyszczone i wstępnie obrobione.
* **MDL\_** - trzecia warstwa, modeled, gdzie dane przybierają ostateczny kształt.

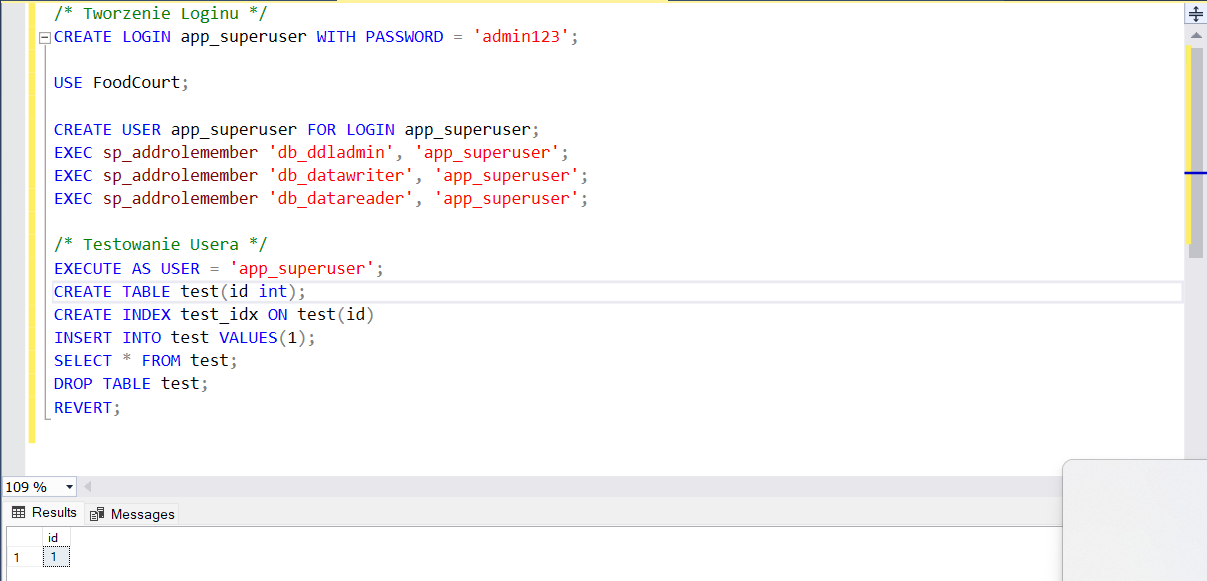
Nasz zespół stanowią bazodanowcy, tak więc do naszych obowiązków należy przygotowanie warstwy **ING\_.**

# Scenariusze I Rozwiazanie

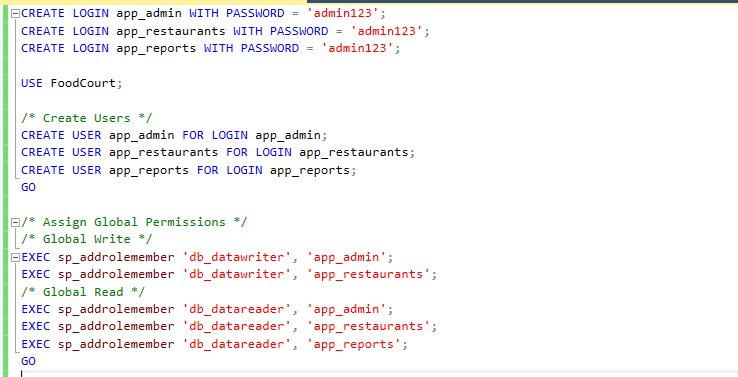
## Tworzenie kont i przypisanie ról systemowych

Wszystkie role, które będą przypisane poszczególnym częściom systemu powinny być tworzone ręcznie. Dodatkowo wszystkie będą miały prefix ‘app\_’.

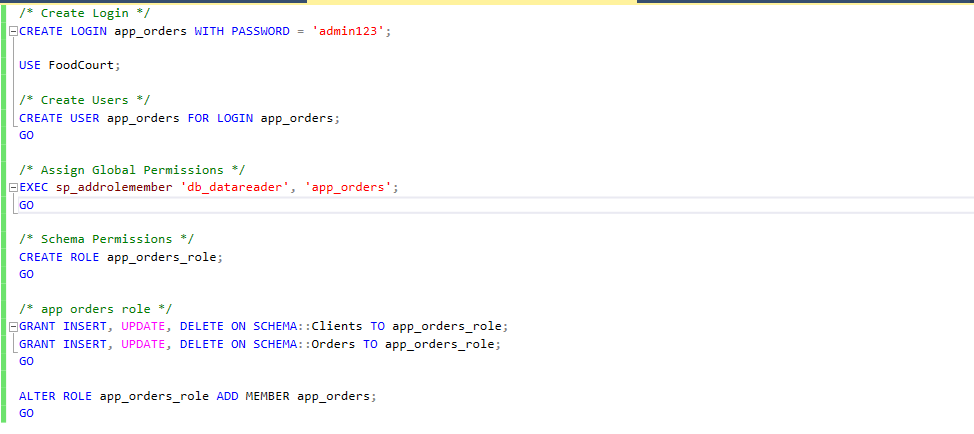
**APP SUPERUSER**



**APP ADMIN, RESTAURANTS, REPORTS**

****

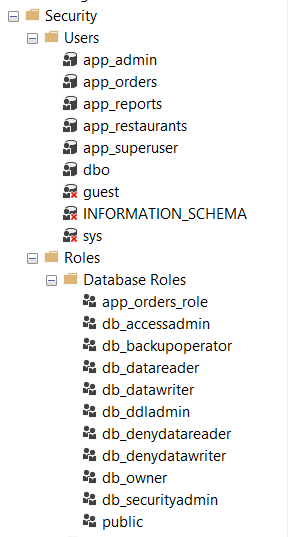
**APP ORDERS**

****

## 

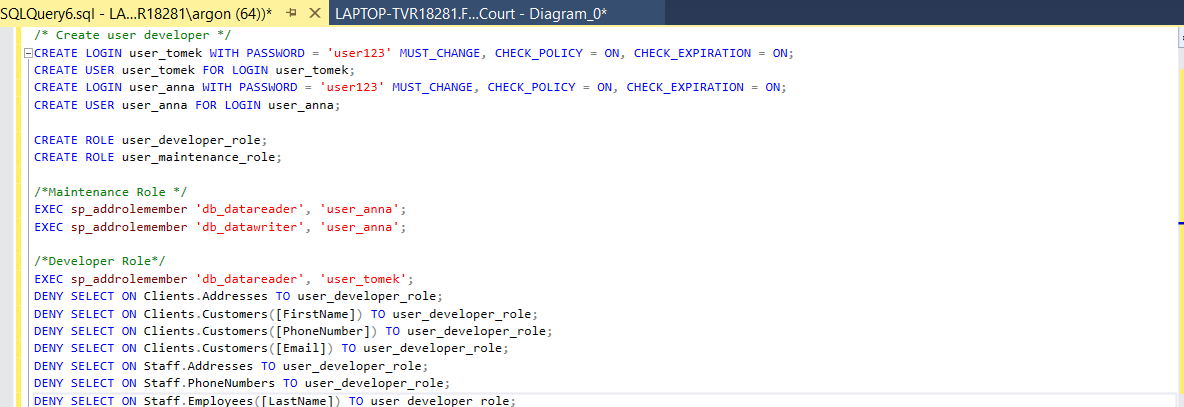
## 

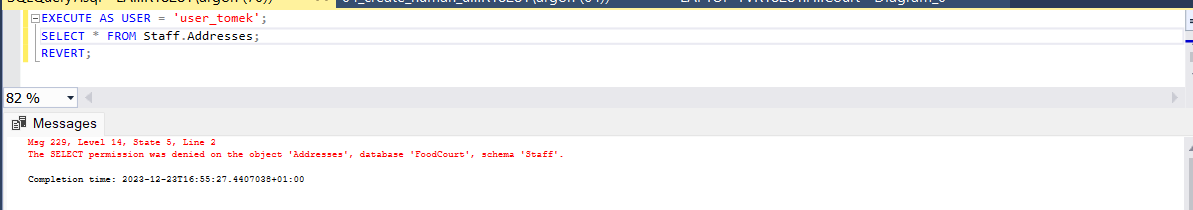
**Podsumowanie**

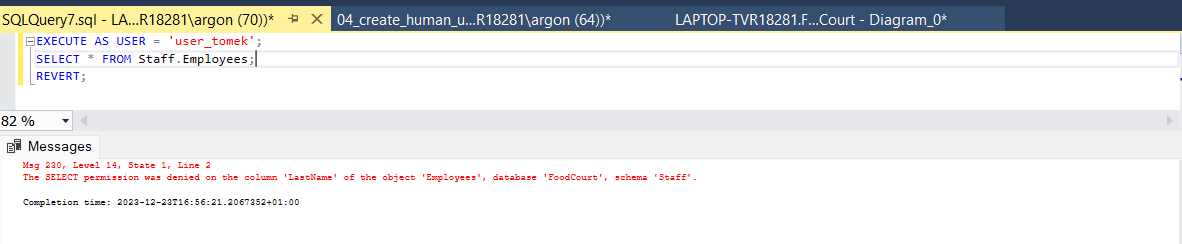
****

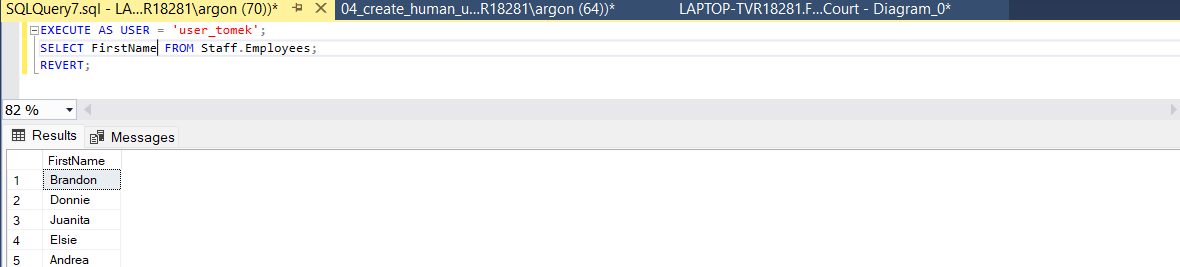
## Tworzenie kont i przypisanie ról użytkowników

Wszystkie role, które będą przypisane poszczególnym częściom systemu powinny być utworzone ręcznie i w zadne sposob nie opierać się na rolach serwerowych. Dodatkowo wszystkie będą miały prefix ‘user\_’.





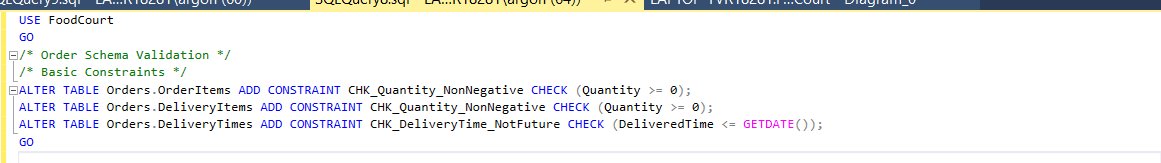


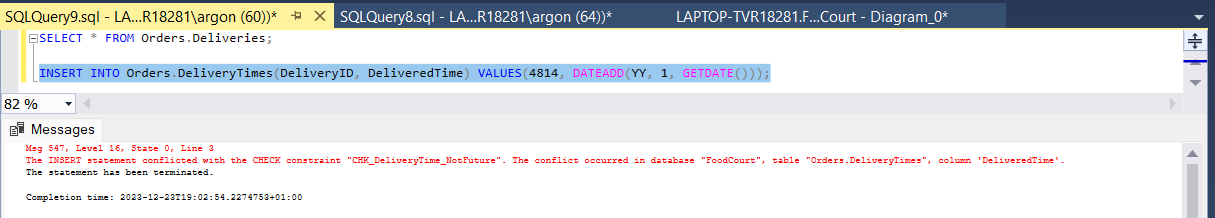


## 

## Walidacja Danych Zamówień

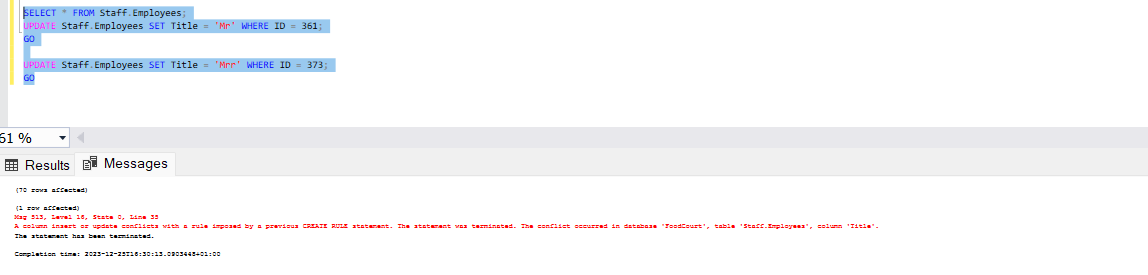
**Base Constraints**



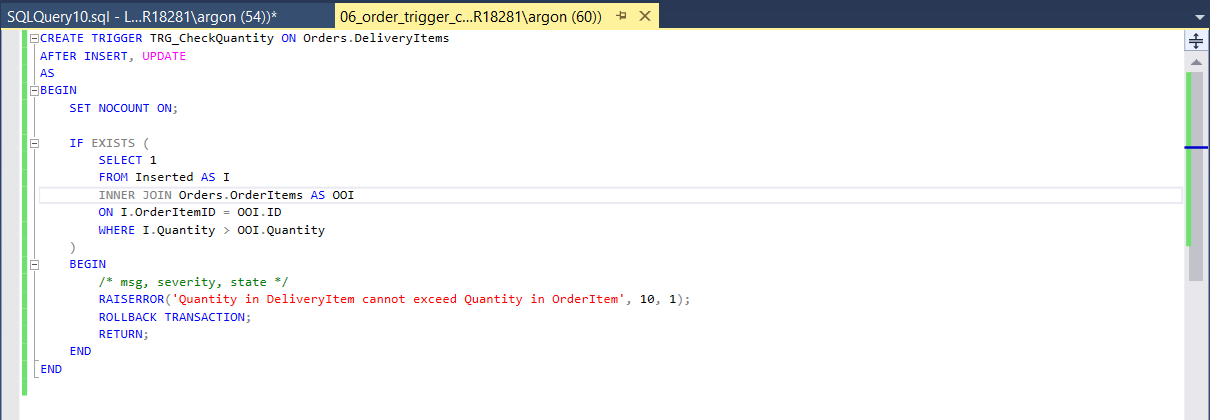


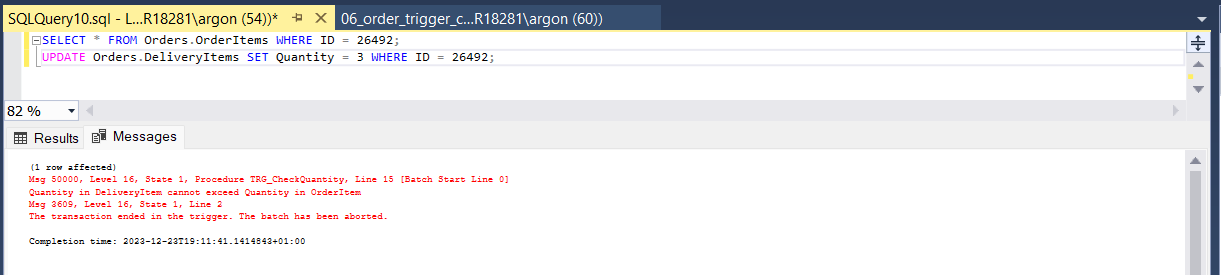
**Base Rules**

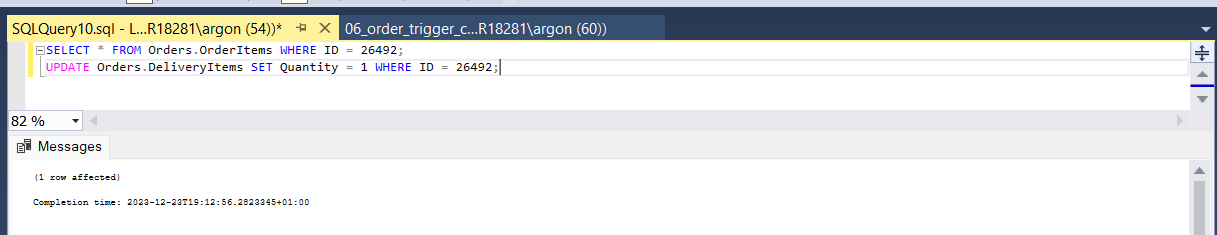
****

****

**Check Triggers**

****

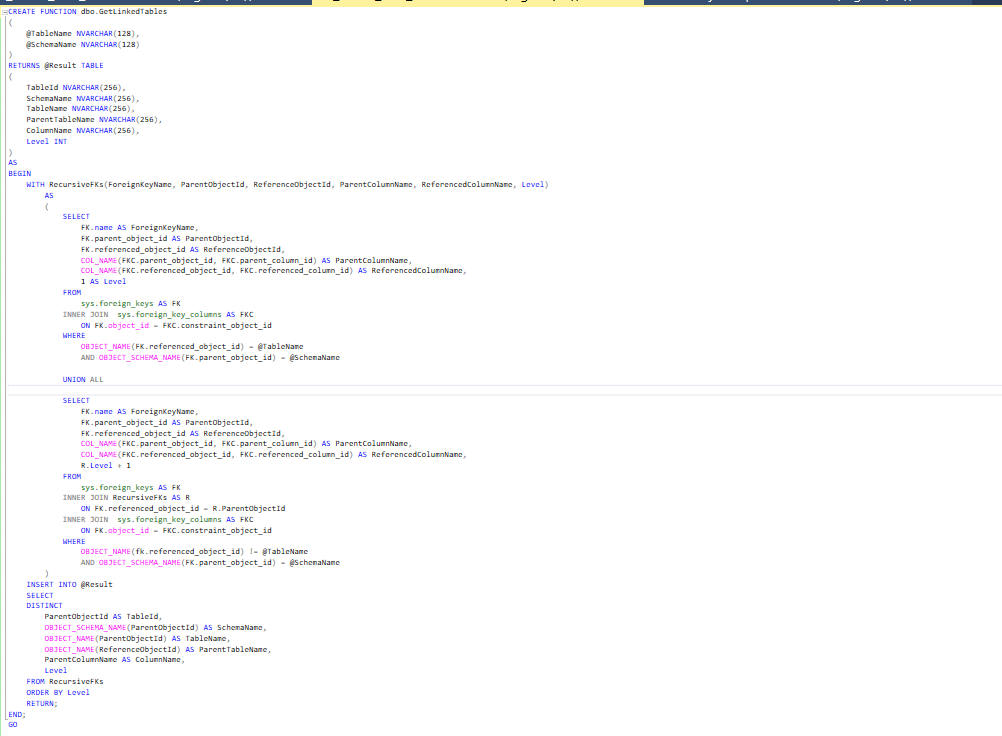




## Mechanizm Soft Delete Zamówień

**Pobieranie Powiązań**

Funkcja tabelaryczna która zwraca dane o powiązaniach bazując na nazwie tabeli i schemie w której ma operować.

****

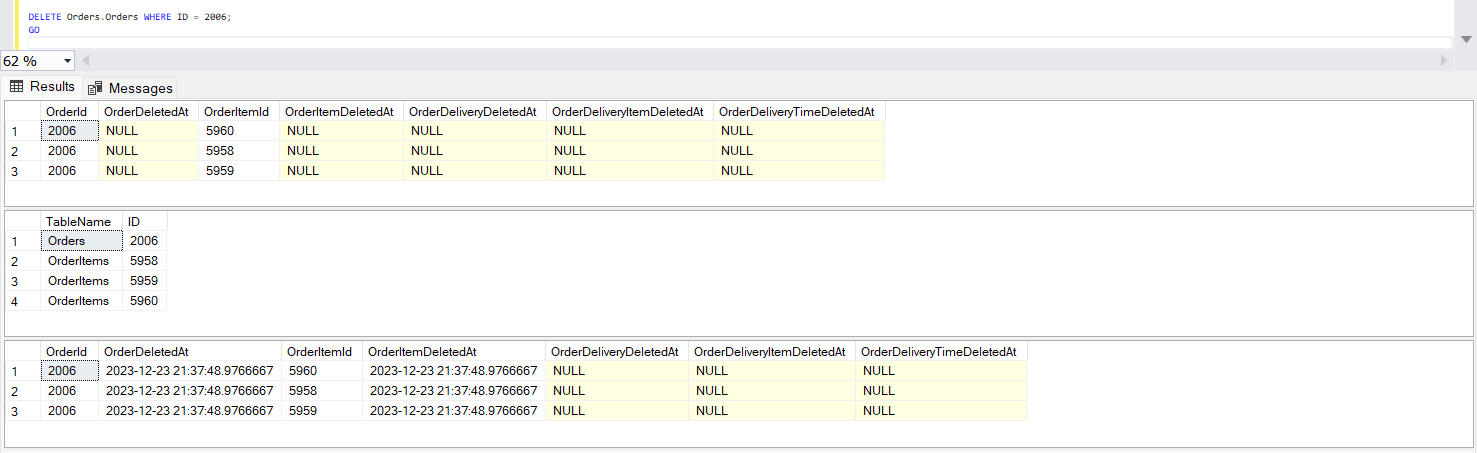
**Mechanizm Soft Delete**

Trigger, który zamiast usuwać Order, dodaje wartość to pola deleted at i uzupełnia kolejne tabele powiązane.

****

**Sprawdzenie**

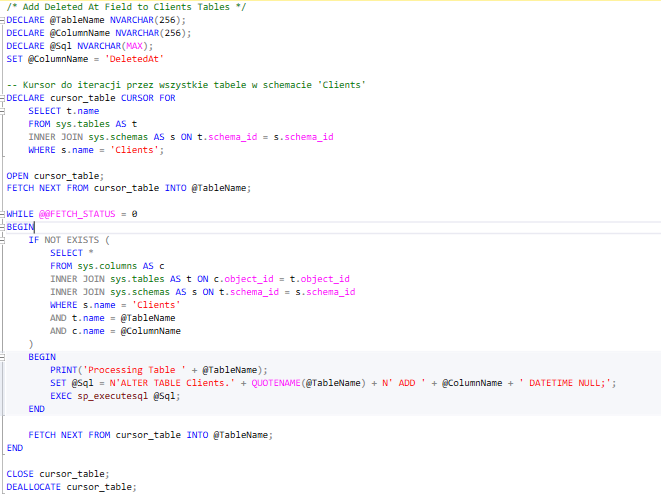
* Pierwsze okno przed usunięciem
* Drugie okno wybranie wartości
* Trzecie okno po usunięciu

****

## Mechanizm Anonimizowania Danych Klienta

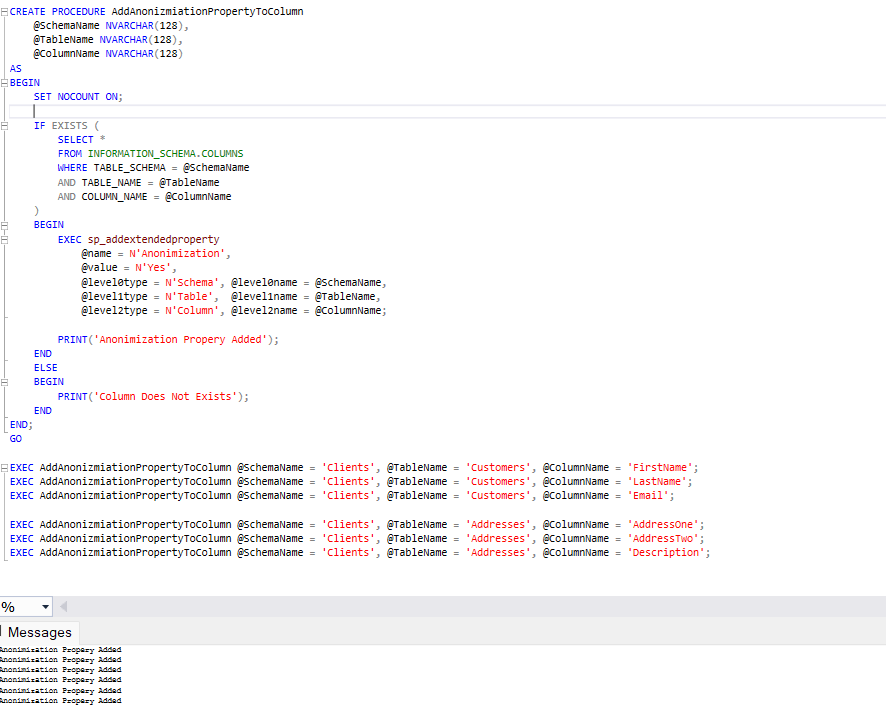
**Dodawanie Pola Deleted At**

Dynamiczne dodanie pola deleted\_at do wszystkich tabel w schemacie Clients.



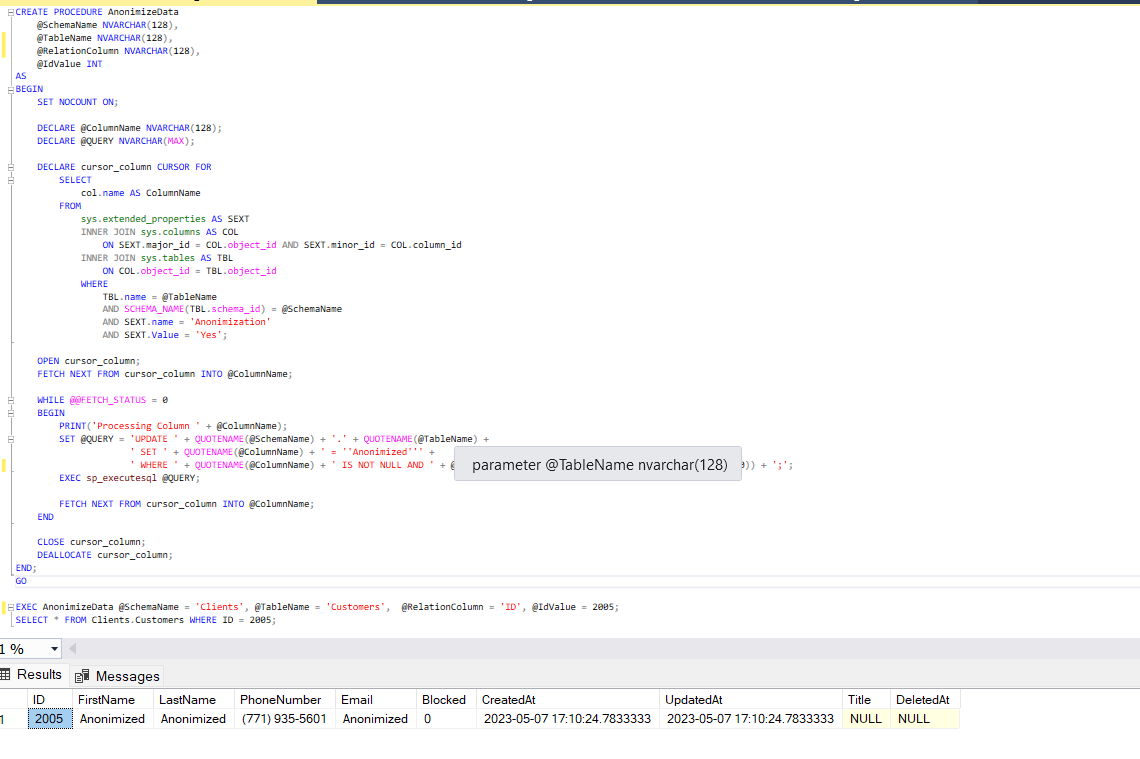
**Oznaczenia Kolumn Do Anonimizacji**

Oznaczanie kolumn do anonimizacji, to powinna być odpowiedzialność DB Admina, lub developera by określić które należy anonimizować, a które nie.

****

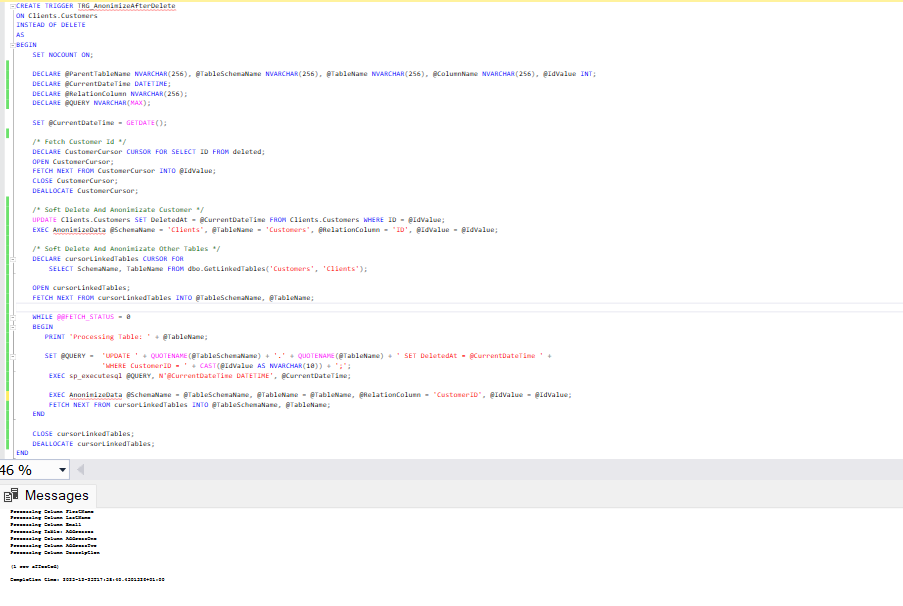
**Dodawanie Procedury Anonimizujące**

Dodanie procedury anonimizującej, która korzystająć w powyższych ustawień będzie anonimizować wybrane pola.

****

**Dodawanie Triggera Anonimizujące**

Dodanie triggera, który po usunięciu rekordu zanonimizuje wybrane pola

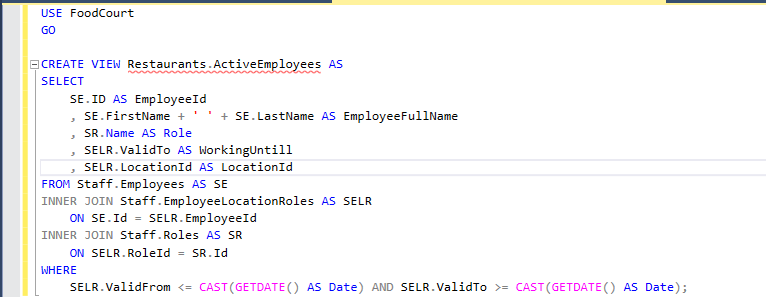
****

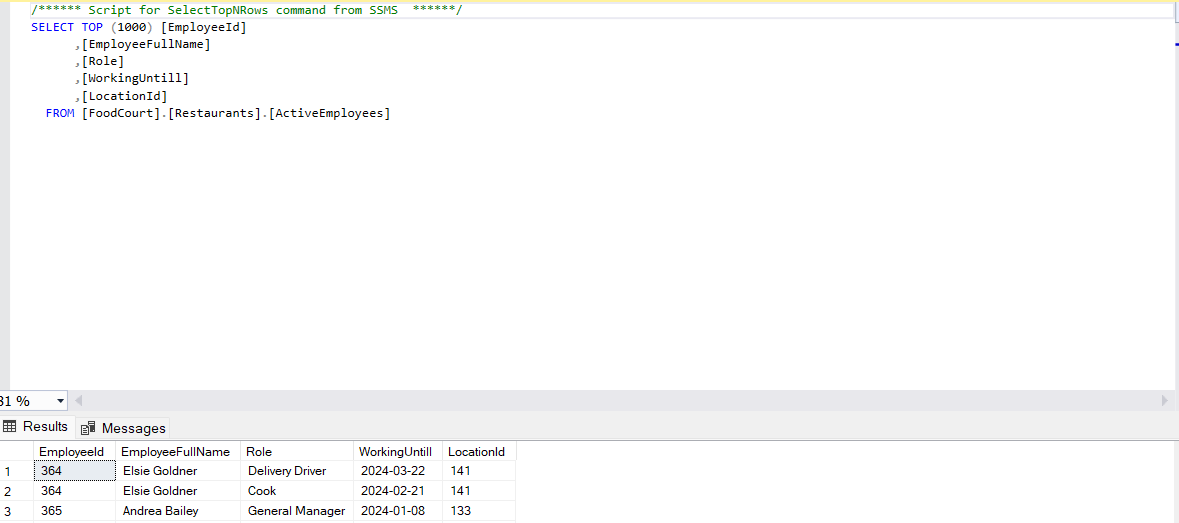
## 

## Widoki bazodanowe

**Widok Aktywnych Pracowników**

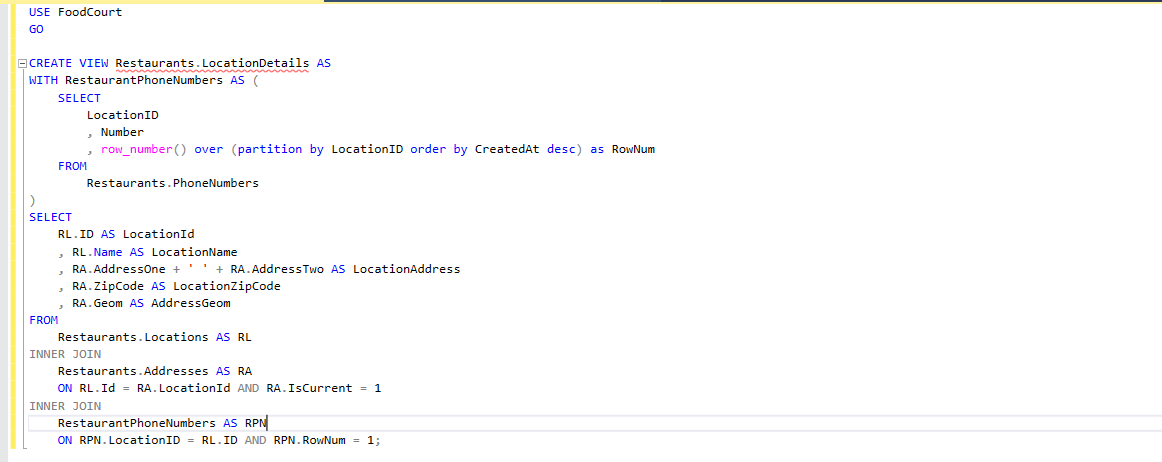
Widok ten zwraca pracowników, którzy aktualnie pracują w danej lokacji.

****

****

**Widok Szczegółowy Lokacji**

Widok ten zwraca aktualne dane adresowe lokacji.

****

## 

## Raportowa Baza Danych

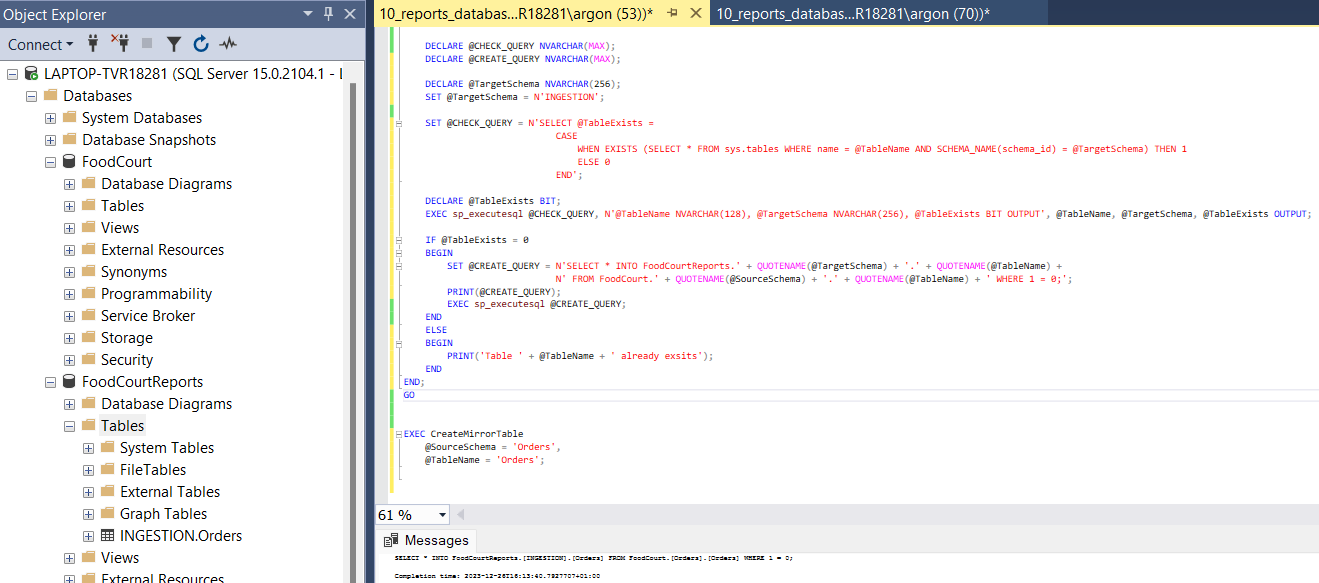
Raportowa baza danych powinna być utrzymywana w formacie ELT, to znaczy że do warstwy ingestion powinny trafiać wszystkie dane z głównej bazy danych. Jako, że rekordy są soft-deletowane, nie musimy przejmować się usuwaniem danych z INGESTION.

**Utworzenie bazy danych i schematu ingestion.**



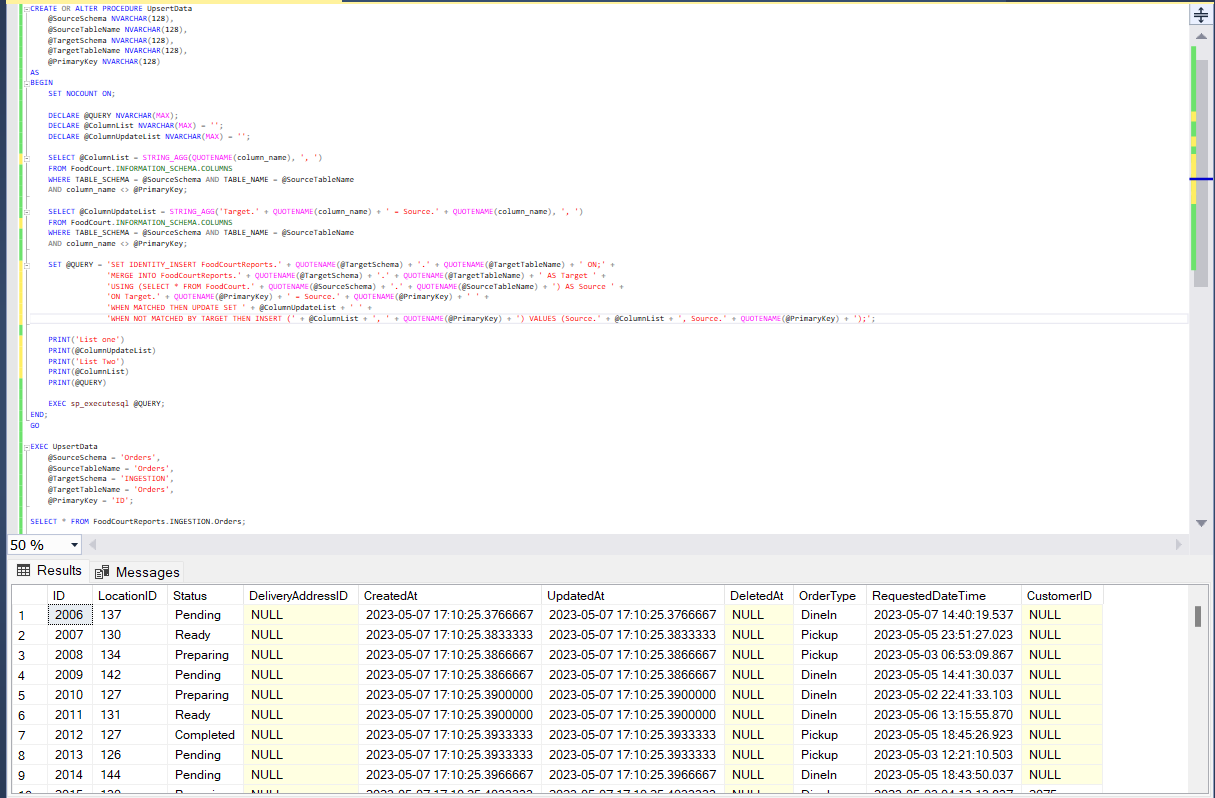
**Utworzenie mirrora tabeli w schemacie Ingestion**

Mirror, lub kopia tabeli powinna odzwierciedlać tabele główną. Możemy się jeszcze w przyszłości pokusić o dodanie podobnej anonimizacji jak w procesie usuwania.

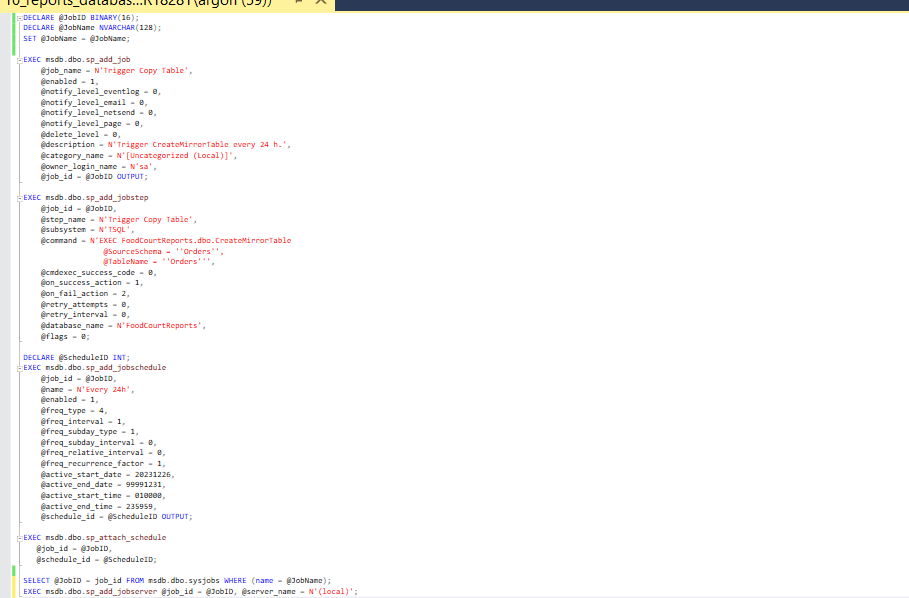
****

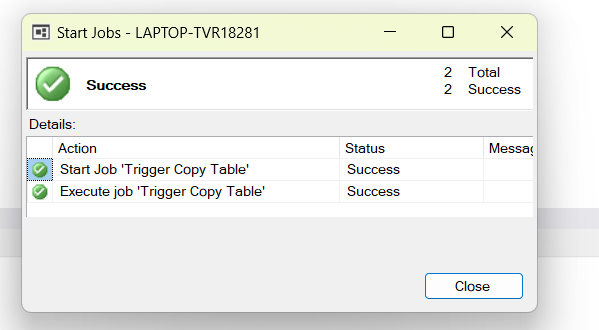
**Przeniesienie danych do Ingestion**

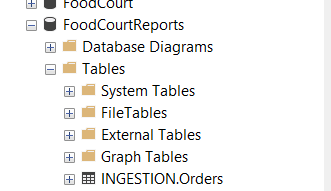
W przypadku ELT, możemy przenosić dane 1-1. W dzisiejszych czasach pamięć jest tańsza niż koszty pracy. ELT zajmuje 2x więcej miejsca ale daje nam możliwość naprawiania danych bez sięgania do zródła.

**Automatyzacja procesu**

Procesy takie powinny być wykonywane automatycznie, bez ingerencji człowieka.

****

****

****

## 

# Podsumowanie i wnioski

Do przygotowania projektu, wykorzystałem poprzedni projekt w którym już zawarte są klucze obce, widoki i schematy bazy danych. Projekt ten załączam jako drugi plik.

W powyższym ćwiczeniu, wiele operacji powinno się wykonać wielokrotnie dla wielu tabel. W skrypcie zamieszczam jednak tylko pojedyncze wykonanie każdego z nich, by nie tworzyć sztuki dla sztuki i dziesiątek kartek do przejrzenia.

Bezpieczeństwo dostępu do danych, jest w dzisiejszych czasach jest jednym z najważniejszych obowiązków inżyniera, ustawienie wielu różnych ról, loginów i kont może nie tylko nam pomóc chronić dane, ale i wyśledzić co poszło nie tak w procesie w przypadku jakiegoś wycieku lub innej formy ataku.

Jeśli chodzi o sekcje pierwszą, z regułami dotyczącymi danych, Rule są ciekawą lecz wydaje mi się trochę przestarzałą opcją. Zdecydowanie wole opierać się na Constraintach lub Enum’ach dla prostej walidacji danych i na Triggerach dla bardziej skomplikowanej logiki biznesowej.

Skrypty są ciekawym i potężnym rozwiązaniem, nie wiem jednak czy pokusiłbym się o szerokie ich stosowanie w komercyjnym projekcie, a jeśli już to na pewno tylko do najważniejszych dla biznesu operacjach, takich jak anonimizacja danych, gdzie błąd na poziomie kodu może kosztować firmę wysokie odszkodowania.

I ostatnie, czyli aplikacja bazodanowa do drugiej bazy danych. Nie znam na tyle możliwości MS SQL, żeby stwierdzić czy rozwiązanie oparte na takich procedurach jest optymalne, ale w prawdziwym życiu wolałbym jednak skorzystać z Kafki jako message broker’a lub SISS w środowisku Microsoft. Ludzie tworzący te rozwiązania zjedli na tym zęby tam gdzie mi rosną dopiero mleczaki. Tym jednak, było to ciekawe zadanie.

## 

# Skrypt

Z racji wielkości plików generujących bazę pliki są załączone oddzielnie i dostępne na [GitHubie](https://github.com/Argonus/codebook/tree/databases-2).